Programming with ASP.NET

Index

Topic-1 : Building ASP.NET Pages

ASP.NET 2.0 is the latest version of ASP, and it represents the most dramatic change yet. With ASP.NET, developers no longer need to paste together a jumble of HTML and script code in order to program the Web. Instead, you can create full-scale web applications using nothing but code and a design tool such as Visual Studio 2005. The cost of all this innovation is the learning curve.

Server-Side Programming

To understand why ASP.NET was created, it helps to understand the problems of other web development technologies. With the original CGI standard, for example, the web server must launch a completely separate instance of the application for each web request. If the website is popular, the web server must struggle under the weight of hundreds of separate copies of the application, eventually becoming a victim of its own success.

Client-Side Programming

At the same time that server-side web development was moving through an alphabet soup of technologies, a new type of programming was gaining popularity. Developers began to experiment with the different ways they could enhance web pages by embedding multimedia and miniature applets built with JavaScript, DHTML (Dynamic HTML), and Java code. These client-side technologies don't involve any server processing. Instead, the complete application is downloaded to the client browser, which executes it locally.

The greatest problem with client-side technologies is that they aren't supported equally by all browsers and operating systems. One of the reasons that web development is so popular in the first place is because web applications don't require setup CDs, downloads, and other tedious (and error-prone) deployment steps. Instead, a web application can be used on any computer that has Internet access. But when developers use client side technologies, they encounter a few familiar headaches. Suddenly, cross-browser compatibility becomes a problem. Developers are forced to test their websites with different operating systems and browsers, and they might even need to distribute browser updates to their clients. In other words, the client-side model sacrifices some of the most important benefits of web development.

ASP.NET allows you to combine the best of client-side programming with server-side programming. For example, the best ASP.NET controls can intelligently detect the features of the client browser. If the browser supports JavaScript, these controls will return a web page that incorporates JavaScript for a richer, more responsive user interface. However, no matter what the capabilities of the browser, your code is always executed on the server.

ASP.NET deals with these problems (and many more) by introducing a completely new model for web pages. This model is based on a remarkable piece of technology called the .NET Framework.

➔ The .NET Framework

You should understand that the .NET Framework is really a cluster of several technologies:

The .NET languages: These include C# and VB .NET (Visual Basic .NET), the objectoriented and modernized successor to Visual Basic 6.0; these languages also include JScript .NET (a server-side version of JavaScript), J# (a Java clone), and C++ with Managed Extensions.

The CLR (Common Language Runtime): The CLR is the engine that executes all .NET programs and provides automatic services for these applications, such as security checking, memory management, and optimization.

The .NET Framework class library: The class library collects thousands of pieces of prebuilt functionality that you can "snap in" to your applications. These features are sometimes organized into technology sets, such as ADO.NET (the technology for creating database applications) and Windows Forms (the technology for creating desktop user interfaces).

ASP.NET: This is the engine that hosts web applications and web services, with almost any feature from the .NET class library. ASP.NET also includes a set of web-specific services.

Visual Studio: This optional development tool contains a rich set of productivity and debugging features. The Visual Studio setup CDs (or DVD) include the complete .NET Framework, so you won't need to download it separately.

ASP.NET File Types

ASP.NET have many types of files. They are,

.aspx : These are ASP.NET web pages (the .NET equivalent of the .asp file in an ASP application). They contain the user interface and, optionally, the underlying application code. Users request or navigate directly to one of these pages to start your web application.

.ascx : These are ASP.NET user controls. User controls are similar to web pages, except that they can't be accessed directly. Instead, they must be hosted inside an ASP.NET web page. User controls allow you to develop a small piece of user interface and reuse it in as many web forms as you want without repetitive code.

.asmx : These are ASP.NET web services. Web services work differently than web pages, but they still share the same application resources, configuration settings, and memory.

web.config : This is the XML-based configuration file for your ASP.NET application. It includes settings for customizing security, state management, memory management, and much more.

global.asax : This is the global application file. You can use this file to define global variables (variables that can be accessed from any web page in the web application) and react to global events (such as when a web application first starts).

.cs / .vb : These are code-behind files that contain C# code or VB code. They allow you to separate the application from the user interface of a web page. **The Page Class :** Every web page is a custom class that inherits from System.Web.UI.Page. By inheriting from this class, your web page class acquires a number of properties that your code can use. These include properties for enabling caching, validation, and tracing.

Property	Description
Application and Session	These collections hold state information on the server.
Cache	This collection allows you to store objects for reuse in other pages or for other clients.
Controls	Provides a collection of all the controls contained on the web page. You can also use the methods of this collection to add new controls dynamically.
EnableViewState	When set to false, this overrides the EnableViewState property of the contained controls, thereby ensuring that no controls will maintain state information.
IsPostBack	This Boolean property indicates whether this is the first time the page is being run (false) or whether the page is being resubmitted in response to a control event, typically with stored view state information (true). This property is often used in the Page.Load event handler, thereby ensuring that basic setup is performed only once for controls that maintain view state
Request	Refers to an HttpRequest object that contains information about the current web request, including client certificates, cookies, and values submitted through HTML form elements. It supports the same features as the built-in ASP Request object.
Response	Refers to an HttpResponse object that allows you to set the web response or redirect the user to another web page. It supports the same features as the built-in ASP Response object, although it's used much less in .NET development.
Server	Refers to an HttpServerUtility object that allows you to perform some miscellaneous tasks, such as URL and HTML encoding. It supports the same features as the built-in ASP Server object.
User	If the user has been authenticated, this property will be initialized with user information

The Control Class : The Page.Controls collection includes all the controls on the current web form. You can loop through this collection and access each control. You can also use the Controls collection to add a dynamic control.

The HttpRequest Class : The HttpRequest class encapsulates all the information related to a client request for a web page. Most of this information corresponds to low-level details such as posted-back form values, server variables, the response encoding, and so on. If you're using ASP.NET to its fullest, you'll almost never dive down to that level. Other properties are generally useful for retrieving information, particularly about the capabilities of the client browser.

Property	Description
ApplicationPath and PhysicalPath	These collections hold state information on the server.
Browser	This collection allows you to store objects for reuse in other pages or for other clients.
ClientCerticate	Provides a collection of all the controls contained on the web page. You can also use the methods of this collection to add new controls dynamically.
Cookies	When set to false, this overrides the EnableViewState property of the contained controls, thereby ensuring that no controls will maintain state information.
Headers and Server Variables	This Boolean property indicates whether this is the first time the page is being run (false) or whether the page is being resubmitted in response to a control event, typically with stored view state information (true). This property is often used in the Page.Load event handler, thereby ensuring that basic setup is performed only once for controls that maintain view state
IsAuthenticated and IsSecureConnection	Returns true if the user has been successfully authenticated and if the user is connected over SSL (also known as the Secure Sockets Layer).
QueryString	Provides the parameters that were passed along with the query string.
Url and UrlReferrer	Provides a Uri object that represents the current address for the page and the page where the user is coming from (the previous page that linked to this page).
UserAgent	A string representing the browser type. Internet Explorer provides the value MSIE for this property.
UserHostAddress and UserHostName	Gets the IP address and the DNS name of the remote client. You could also access this information through the ServerVariables collection.
UserLanguages	Provides a sorted string array that lists the client's language preferences. This can be useful if you need to create multilingual pages.

The HttpResponse Class : The HttpResponse class allows you to send information directly to the client. In traditional ASP development, the Response object was used heavily to create dynamic pages. The HttpResponse does still provide some important functionality, namely, caching support, cookie features, and the Redirect method.

Property	Description
BufferOutput	When set to true (the default), the page isn't sent to the client until it's completely rendered and ready, as opposed to being sent piecemeal.
Cache	References an HttpCachePolicy object that allows you to configure how this page will be cached
Cookies	The collection of cookies sent with the response.
Write(), BinaryWrite(), and WriteFile()	These methods allow you to write text or binary content directly to the response stream. You can even write the contents of a file. These methods are de-emphasized in ASP.NET and shouldn't be used in conjunction with server controls.
Redirect()	This method transfers the user to another page in your application or a different website.